Haunted by Legacy: Discovering and Exploiting Vulnerable Tunnelling Hosts

Angelos Beitis
DistriNet, KU Leuven
angelos.beitis@kuleuven.be

Mathy Vanhoef
DistriNet, KU Leuven
Mathy.Vanhoef@kuleuven.be

Abstract

This paper studies the prevalence and security impact of open tunnelling hosts on the Internet. These hosts accept legacy or modern tunnelling traffic from any source. We first scan the Internet for vulnerable IPv4 and IPv6 hosts, using 7 different scan methods, revealing more than 4 million vulnerable hosts which accept unauthenticated IP in IP (IPIP), Generic Routing Encapsulation (GRE), IPv4 in IPv6 (4in6), or IPv6 in IPv4 (6in4) traffic. These hosts can be abused as one-way proxies, can enable an adversary to spoof the source address of packets, or can permit access to an organization's private network. The discovered hosts also facilitate new Denial-of-service (DoS) attacks. Two new DoS attacks amplify traffic: one concentrates traffic in time, and another loops packets between vulnerable hosts, resulting in an amplification factor of at least 16 and 75, respectively. Additionally, we present an Economic Denial of Sustainability (EDoS) attack, where the outgoing bandwidth of a host is drained. Finally, we discuss countermeasures and hope our findings will motivate people to better secure tunnelling hosts.

1 Introduction

Tunnelling protocols are an essential backbone of the Internet, with examples being the IP in IP (IPIP) and the Generic Routing Encapsulation (GRE) protocol [37, 49]. These protocols can link disconnected networks and form Virtual Private Networks (VPNs). One limitation of these protocols is that they do not authenticate or encrypt traffic. Instead, to secure these protocols, they must be combined with Internet Protocol Security (IPsec) [53].

Unfortunately, tunnelling protocols are often used without extra security, allowing an on-path or even off-path attacker to inject traffic into the tunnel [38, 64]. More troubling is how recently Yannay has demonstrated that misconfigured IPv4 hosts may accept unauthenticated IPIP tunnelling traffic from any source [39]. He also showed that an adversary can abuse these vulnerable hosts to spoof IPv4 addresses. Although this was an excellent discovery, several questions remain unanswered. In particular, it is unclear: (1) whether IPv6 hosts can also be vulnerable; (2) how to best scan for vulnerable hosts; (3) whether other tunnelling protocols can be vulnerable; (4) what the security implications of vulnerable tunnelling hosts are; and (5) what some practical defences are.

To answer these questions, we systematically analyse tunnelling protocols, including GRE, IP6IP6, Generic UDP Encapsulation (GUE), IPv4 in IPv6 (4in6), and IPv6 in IPv4 (6in4), and devise 7 different scanning methods to detect hosts that accept unencrypted tunnelling traffic from any source. We use these methods to scan the Internet for vulnerable IPv4 and IPv6 hosts. This reveals that all studied tunnelling protocols can be vulnerable: in total, we detected 3,527,565 vulnerable IPv4 hosts and 735,628 vulnerable IPv6 hosts.

Even more problematic is that many vulnerable hosts can be abused to spoof the source address of packets. This is significant since the increased adoption of source address filtering was thought to make spoofing harder for ordinary Internet users [6]. Vulnerable hosts may also allow access to an organisation's private network, where carefully constructed tunnelling packets can reveal details about the victim's (private) network, facilitating subsequent attacks. Additionally, we present an administrative Denial-of-service (DoS) attack, where traffic is spoofed to trigger incorrect abuse reports towards the victim.

We also investigate how vulnerable hosts can facilitate new DoS attacks. We uncover two new amplification DoS attacks, called *Tunnelled-Temporal Lensing (TuTL)* and the *Ping-Pong* attack, with an amplification factor of at least 16 and 75, respectively. Additionally, we present an Economic Denial of Sustainability (EDoS) attack, where the outgoing bandwidth of a host is consumed to incur unexpected financial costs or lead to service disruptions.

To summarise, our main contributions are:

- We scan the Internet for vulnerable hosts that accept tunnelling packets from any source (Section 3 and 4).
- We present a new administrative DoS attack (Section 4).
- We introduce three tunnelling-based DoS attacks: Ping-

Pong and Tunnelled-Temporal Lensing (TuTL) amplification, and an EDoS attack (Section 5).

- We present defences against our new attacks (Section 6).

Finally, we discuss and compare to related work in Section 7, and we conclude in Section 8.

2 Background

This section explains tunnelling protocols, discusses known issues with IPIP, and introduces temporal lensing DoS attacks.

2.1 Tunnelling Protocols

A tunnel in the context of the Internet is a protocol to transfer frames or packets between two (disconnected) networks [52]. Generally, a tunnelling protocol encapsulates packets and sends them to the receiver. The receiver will, in turn, strip possible headers and acquire the original encapsulated packet, which can then be forwarded [52]. Tunnelling protocols are usually utilised to transport layer two frames or layer three packets [49, 62]. Tunnels have several applications and can be used for different reasons depending on the use case.

In this paper, we will discuss tunnelling protocols that do not, by default, use authentication or encryption and demonstrate how attackers can use hosts that implement such tunnelling interfaces to spoof and conduct DoS attacks. We will use the term *vulnerable hosts* to refer to hosts that implement a tunnelling protocol insecurely, meaning that they accept plaintext tunnelling traffic from arbitrary sources. We define the term *host* as a synonym for an allocated IPv4 or IPv6 address. This implies that a single physical machine might represent multiple hosts if it has multiple (virtual) network cards. The term *unfiltered networks* refers to networks that do not implement source address filtering [6]. In other words, hosts in an unfiltered network can transmit IP packets using an arbitrary source IP address.

2.1.1 IPIP & IP6IP6. The IPIP tunnelling protocol was standardised in 1996 as RFC 2003 [49], and the IP6IP6 protocol was standardised in 1998 as RFC 2473 [12]. These protocols have no authentication or encryption and do not add any additional headers when transporting an inner packet. In other words, with IPIP, the IPv4 header of the outer (tunnelling) packet is directly followed by the inner (encapsulated) IPv4 packet, and similarly for IP6IP6 with IPv6 packets.

2.1.2 GRE & GRE6. The GRE protocol introduced a more general approach to encapsulation, which was standardised as RFC 2784 in 2000 [37]. Unlike IPIP, the GRE protocol adds a custom header before encapsulating the (inner) packet. This allows GRE to encapsulate other EtherType protocols, such as IPv4 packets and IPv6 packets. Furthermore, the GRE header has been extended in RFC 2890 and can include an optional 4-byte key field [15]. When this key field is used, the recipient will only accept a GRE packet containing the expected key value. Unfortunately, this key field is not encrypted and can be trivially eavesdropped by an adversary. The key field is optional and remains identical unless explicitly changed, offering little security in practice.

Although the GRE key field does not prohibit a man-in-the-middle attacker from spoofing packets, it does make it harder for a man-on-the-side adversary to construct valid packets. In particular, when scanning for vulnerable GRE hosts in Section 3, hosts that use a key value are not detected.

2.1.3 6in4 & 4in6. The *6in4* and *4in6* protocols refer to the encapsulation of an IPv6 packet in an IPv4 packet (6in4) and the encapsulation of an IPv4 packet in an IPv6 packet (4in6) and are defined in RFC 4213 [20] and 2473 [12], respectively. These two protocols were designed to allow IPv4 traffic to flow in an IPv6 network and vice versa, thus allowing interfaces with no IPv6 support to communicate with IPv6 interfaces. The receiver of the datagram will strip the outer IPv4 (for 6in4) or IPv6 (for 4in6) header and serve the packet.

The term *6to4* refers to a transition mechanism for 6in4 packets [8]. Unlike the standard 6in4 transition mechanism, 6to4 introduces its own IPv6 address, which is derived from the host's IPv4 address. The 6to4 address has the format `2002:IPV4_ADDRESS_IN_HEX::`. For example, a 6to4 address of `a.b.c.d`, is `2002:a.b.c.d::`. For the remainder of this paper, we will use the term *6in4* to refer to IPv6 packets encapsulated in IPv4 packets and *6to4 address* to refer to addresses of the aforementioned format.

2.1.4 GUE. GUE can encapsulate different protocols within UDP datagrams [27]. It has two variants, one where a custom GUE header is introduced and one where IPv4 and IPv6 packets can be directly encapsulated. When setting up a GUE tunnel, both parties must specify a UDP port number and the tunnelling protocol responsible for processing the encapsulated packets [26]. Although GUE is currently a draft standard, it is supported by Linux. The optional GUE header is encapsulated inside a UDP frame with the destination port of the decapsulator's UDP interface [27]. After the GUE header, the next encapsulation protocol header follows, which can be any IP protocol.

2.2 Known Weaknesses in IPIP

The IPIP protocol does not provide authentication or confidentiality [49], making it trivial for a man-in-the-middle attacker to inject traffic into an IPIP tunnel. To prevent such attacks, it is recommended to use IPIP in combination with protocols such as IPsec [53]. However, Yannay has recently demonstrated that many IPv4 hosts accept bare IPIP packets without source address filtering [39, 49]. In other words, these vulnerable IPv4 hosts accept plaintext IPIP packets from any sender. As a result, an attacker can send an IPIP packet to a

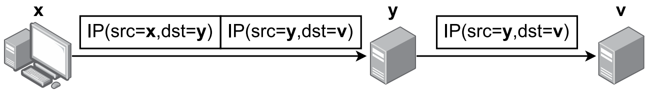


Figure 1: Illustration of Yannay's IPIP spoofing attack [39].

vulnerable host, which will strip the outer header and forward the inner IP packet without authenticating the sender.

An example of Yannay's IPIP attack is shown in Figure 1, which contains an attacker with IP address x, a vulnerable host with IP address y and a victim with IP address v. The IPIP packet sent by an adversary will have two IP headers: the *outer* IP header, which will have address x as source and address y as destination and the *inner* IP header, which will have address y as source and address v as destination. The vulnerable host will receive the IPIP packet, strip the *outer* IP header, and forward the *inner* packet to its destination. Since the source address y of the inner IP header equals the address of the vulnerable host, it does not violate any network filtering policies that might be in place, thus allowing an adversary to send spoofed traffic to a victim [39]. The vulnerable host is then considered a *one-way proxy* for the attacker.

The attacker is not constrained in using address y as the source address of the *inner* header since, in the scenario where the vulnerable host resides in a network with no (or bad) filtering mechanisms, multiple spoofed addresses can be utilised by an adversary. For this scenario to work, an adversary must identify a network with poor filtering that allows a packet with a spoofed source address to be forwarded [39].

2.3 Temporal Lensing attacks

A *temporal lensing attack* refers to a DoS amplification attack, which can enhance an attacker's damage towards the victim [50]. Unlike traditional amplification attacks where the traffic itself is amplified, meaning that an attacker can generate g packets and p packets are received by the victim where $p > g$, in temporal lensing, an amplification effect is achieved by concentrating packets in time. For instance, the attacker sends packets for 10 seconds and uses protocol properties to ensure they arrive at the victim in a window of less than one second, resulting in an amplification factor of at least 10.

The original temporal lensing by Rasti et al. abused Domain Name System (DNS) resolvers, where an attacker could construct different paths with different latencies towards a target DNS server [50]. The attacker will then schedule packets over these paths and cause them to arrive at the victim at practically the same time.

3 Internet Scanning

In this section, we describe how we selected the tunnelling protocols to scan for, explain how we first did a preliminary analysis of each selected protocol, describe the different scanning methods we created to discover vulnerable hosts and clarify how we tackled the scanning of the IPv6 address space.

3.1 Methodology

3.1.1 Protocol Selection. We first identified tunnelling protocols that may cause a host implementing them to accept traffic from arbitrary sources. For a vulnerable host to satisfy this criteria, the tunnelling protocol must have the following properties: (1) no default authentication and (2) no default encryption. Additionally, to avoid testing historic protocols that are no longer in use, we require that the tunnelling protocol is supported by a recent Linux kernel. Going over the list of assigned IP protocol numbers and a subset of the transport protocol port number registry for UDP led us to select IPIP, GRE, GUE, 4in6, and 6in4 for study [12, 20, 27, 37, 49].

3.1.2 Preliminary Experiments. Before scanning the Internet for vulnerable hosts, we performed local experiments to confirm that vulnerable hosts may exist for each selected tunnelling protocol. For these experiments, we set up two instances which would take the roles of attacker and victim. We experimented with these instances to determine if a victim host can be configured to accept tunnelling protocols from an arbitrary source. Concretely, we set up a tunnelling interface on the victim host and then transmit a tunnelling packet from the attacker where the inner header's source address is the victim and the inner destination address is the attacker. We can then determine whether the victim is vulnerable by seeing if the attacker's machine receives the inner packet. Our experiments showed that all selected protocols, i.e., IPIP, GRE, GUE, 4in6, and 6in4, can be configured to accept traffic from an arbitrary source and therefore be vulnerable.

For Linux, we found that the following two factors influenced whether a host was vulnerable:

- *Decapsulating Arbitrary Traffic*: When setting up a tunnelling interface, there is an option to declare a `remote address` [31]. This field allows the user to specify an address or set of addresses from which the interface will decapsulate and serve packets. Additionally, this field can be left empty, which causes the host to decapsulate and serve packets from any source address. In the case of GRE, leaving this field empty effectively creates a multipoint GRE (mGRE) tunnel [58].
- *Forwarding Traffic with Source IP*: When the *remote address* field is not empty, the attacker can still not force the vulnerable host to spoof packets as long as the host's network implements filtering. Furthermore, the Linux kernel will, by default, not forward packets whose source address is that of its own interface [59]. This can be bypassed by setting the `accept_local` variable to True, allowing any encapsulated header with the vulnerable host's address as the source to be forwarded.

Table 1: Scan methodology summary for each protocol.

Methodology	*IPIP*	*IP6IP6*	*GRE*	*GRE6*	*4in6*	*6in4*	*GUE*
Standard	✓⃝	✓	✓	✓	✓	✓	✓
Echo/Reply	✓	✓	✓	✓	–	✗⃝	✗
TTL Expired	✓	✓	✓	✓	✓	✓	✗

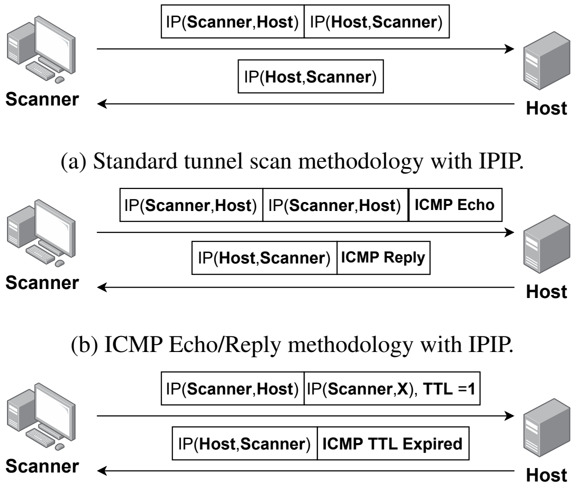
✓ We did this scan. ✗ We did not do this scan.
○ The scan has been done by related work [34, 39].
– The scan method is not possible for the protocol.

When the above two conditions are met, a Linux host is vulnerable, and an adversary can carry out various attacks against the host, as explained in the remainder of the paper.

Before performing a full (standard) Internet scan for each tunnelling protocol, we first scanned 1% of the IPv4 addresses. For GUE, a destination UDP port needs to be specified, and we scanned with port number 6080 as the destination port. This is the default port for GUE and hence has the highest likelihood of detecting vulnerable hosts [27]. However, in our 1% scan, no vulnerable GUE hosts were identified. As a result, we omitted GUE in our (other) Internet-wide scans. To more firmly confirm this, we later further scanned 18% of the IPv4 space for GUE, with again no replies received. For all other protocols, a partial 1% standard scan did reveal vulnerable hosts, and for these protocols, we scanned the entire IPv4 address space using multiple different scanning methods.

3.2 Scanning Methods

Our scanning methodology consists of 7 different scans, each identifying vulnerable hosts with different setups. These scans fall in one of three scanning methods shown in Figure 2, namely *Standard Scans*, *Internet Control Message Protocol (ICMP) Echo/Reply Scans* and *ICMP Time to Live (TTL) Expired Scans*. In the remainder of the paper, Internet scanning refers to sending *probe packets* and accumulating data based on the replies [10]. We will use ZMap and its IPv6 extension, ZMapv6, to scan the Internet [10, 60]. We implemented different *probe modules*, allowing us to identify multiple types of vulnerable tunnelling hosts. Since scanning the IPv6 address space by brute force is infeasible [43], we used an IPv6 hitlist, which lists the latest responsive IPv6 addresses [57]. The scanning methodology remains the same as with IPv4 scans, with the only difference being that modules created with the ZMapv6 extension [60] are given the IPv6 hitlist as input. ZMapv6 will then send probe packets to the IPv6 addresses specified in the histlist. We conducted the scans from two AWS instances in the US (AS14618) and a Time4VPS instance in Lithuania (AS212531). Table 1 summarises each selected protocol, indicating which scans have (not) been conducted by us, as well as those performed by related work [34, 39].



(a) Standard tunnel scan methodology with IPIP.

(b) ICMP Echo/Reply methodology with IPIP.

(c) ICMP TTL Expired methodology with IPIP.

Figure 2: Illustration of the three main scan methodologies. The first arrow shows the scanner's sent probe, while the second arrows show the host's response.

3.2.1 Standard Tunnel Scans.

Firstly, we introduce simple probe packets, which consist of an IP header (version 4 or 6) followed by the protocol to be tunnelled. For IPIP and IP6IP6, the probe consists of an encapsulated IPv4 and IPv6 header, while for GRE and GRE6, the structure of the probe is similar, with an additional GRE header between the two IP headers. When a potentially vulnerable host receives such a packet, its header will be removed (along with the GRE frame in the case of GRE), and the inner header will be forwarded to its destination address. For example, in Figure 2a, a scanning machine wants to identify if a specific host is vulnerable. The scanner will first send the IPIP packet to the host. The host will decapsulate the outer header, and since the remaining header is destined for the scanner, the host will forward it to the destination address. This standard scan allows us to identify addresses that will serve and forward incoming traffic on their tunnelling interface. In this initial scanning methodology, the inner header's IP source address is that of the potentially vulnerable host. This will allow the forwarded traffic to pass potential filtering mechanisms set up to disallow IP source address spoofing. Yannay showcased the methodology in the initial IPIP scanning iteration [39], which in 2020 identified 150k vulnerable IPIP hosts [40].

Subnet-Spoofing Scan. As shown by our preliminary Linux experiments, a host may not forward the inner packet if its source IP address equals the host's address. To nevertheless try to detect such vulnerable hosts, we performed a scan with the Standard tunnel scan methodology, but with the source IP address of the inner packet being in the same subnet as the host's address. This source address was generated by changing the last octet of the host's IP address, ensuring it belongs to the same /24 subnet in IPv4 and the same /112 subnet in IPv6. For instance, if the host has IP address `a.b.c.5`, the inner packet would have source address `a.b.c.6`.

To detect numerous vulnerable hosts, the inner packet's source address must be in the same (smallest) subnet as the host. This is because Autonomous Systems (ASs) might only filter packets before they exit the AS and not within the AS. Filtering can be implemented using techniques such as ingress address filtering as defined in RFC 2827 and 3704 [5,54]. For example, a router with the address `a.b.c.0/24` should reject outwards traffic from the source address `a.b.f.5`, where `c` $\neq$ `f`, since it belongs to a different subnet.

Spoofing Scan. We can make a host forward a packet with a spoofed source IP address, allowing us to identify ASs with poor filtering. The methodology remains the same as in Figure 2a, but with the inner packet's source address being spoofed, e.g., if the host's address is `a.b.c.d`, the inner packet's source address will be `e.f.g.h`. If we receive the inner packet as a response, the host does not correctly implement source address filtering for outgoing packets, meaning the host can be abused to spoof IP packets with an arbitrary source address.

Furthermore, the spoofing scan is also possible with the 6in4 and 4in6 protocols by having a random IPv6 or IPv4 address as the inner packet's source, respectively.

In our scans, we used `100.200.a.b` as the spoofed source.[1]

6to4 & IPv4-Mapped Address Scan. One challenge is scanning for mixed IPv4 and IPv6 protocols such as 6in4 and 4in6. For instance, in a 6in4 scan, we do not know the IPv6 address of a host when probing through its IPv4 address. This complicates conducting the standard tunnel scans. Thankfully, in the case of 6in4, we can do a standard scan where the source address of the inner IPv6 packets equals the *6to4* or *IPv4-mapped* address of the host. Both 6to4 and IPv4-mapped IPv6 addresses are derived from the host's (outer) IPv4 address. The *IPv4-mapped* address has a similar format to the 6to4 address, shown in Section 2. The IPv4-mapped format is `::ffff:IPV4_ADDRESS_IN_HEX` [13]. This allows us to let a vulnerable host forward an IPv6 packet towards the scanner with a valid source IPv6 address. Since the packet has a valid IPv6 source address, it is less likely to be filtered or blocked by the host's network. This methodology further allows us to identify 6to4 relay routers [34].

3.2.2 Tunnelled ICMP Echo/Reply Scan.

The previous scanning methods can detect hosts that accept tunnelling packets *and* that forward the inner packet. However, in practice, not all hosts may be configured to forward the inner packet. To circumvent this limitation and identify vulnerable hosts that cannot forward traffic, we let the inner packet be an ICMP Echo request to the host itself, as shown in Figure 2b. In other words, the packet's inner header will have the scanner's address as the source and the host's address as the destination. Upon receiving the packet, a vulnerable host will strip the outer header, serve the inner Echo request to itself, and finally respond with an ICMP Echo reply to the scanner. A similar scanning method has been previously used by Kristoff et al. to discover open 6to4 relays [34]. In their study, they discovered 1.5 million IPv4 addresses. Therefore, we decided to exclude this method for 6in4 scans since it was previously conducted by related work. This method is infeasible for 4in6 scans since it is impossible to map an IPv6 address to an IPv4 one.

3.2.3 Tunnelled ICMP TTL Expired Scan.

Not all hosts are configured to reply to ICMP Echo requests and, therefore, cannot be discovered by the previous method. Additionally, some hosts may be located behind private networks. For example, some Virtual Private Cloud (VPC) providers allocate a private address to their instances, and any traffic towards their public IP will be translated to the private IP of the instance, i.e., inbound Network Address Translation (NAT) may be used. Moreover, these VPCs may only allow outgoing packets when the packet's source address equals the private IP address of the host. As a result, a standard or subnet spoofing scan would also not detect such vulnerable hosts because the inner packet will not use the host's private IP address and, hence, will be dropped by the VPC.

To overcome this complication, we can send an inner IPv4 header with a TTL equal to one or an IPv6 header with a hop limit equal to zero and the scanner as the source address. The inner packet's destination can be any address other than the host's. For example, in Figure 2c, the vulnerable hosts that receive the probe will pass the inner packet to its tunnelling interface. When validating the header's TTL value, it will detect that the TTL has expired and, as a result, discard the inner IP header, sending an ICMP or ICPMv6 TTL Expired message to the inner packet's source address. The scanner will then receive the reply, and based on the ICMP TTL Expired payload, which is the expired IP header along with 64 bits of the original datagram [51], can confirm that the inner IP header has expired in transit.

3.3 Ethics

To ensure our scans have no negative impact, we limit the daily number of packets sent to a single address, guaranteeing that scanned hosts never get overloaded. We limit ZMap's scanning rate to 10-20MBs, meaning IPv4 scans take roughly 4 days to complete. Setting a low scanning rate also ensures that networks do not receive overwhelming amounts of traffic. For IPv6 scans, we conducted a scan only once a day.

All probes have a benign ICMP packet as payload. The spoofed ICMP Echo reply is necessary to understand how many vulnerable hosts can spoof their source address. This

[1] The spoofed IP address has been partially anonymised to ensure our research does not negatively affect its reputation or normal usage.

ICMP reply has our own server as its destination, which isolates the spoofed traffic between the scanning server (us) and the host being scanned. The spoofed ICMP Echo reply is also kept as benign as possible: it resembles a normal and short packet, which is unlikely to be considered malicious by intermediate devices on the path between the vulnerable host and our scanning server. To limit the number of spoofed ICMP Echo replies that a vulnerable host would generate and reduce the amount of traffic it would receive, we decided to spoof only a single source address. This already gives us sufficient information to estimate how many hosts can spoof source addresses. We contacted the owner of the spoofed IP to ensure that our spoofed scans had not caused any negative effects.

We also follow standard practice and host a website on the scanning server with information about the scan [10, 16]. The website mentioned the scan dates, methodologies, scanned protocols, and our email addresses in case any organisation wanted to opt out of any future scans. Our scanning code also included a reference to our scanning server in the ICMP packet. This means that in the unlikely circumstance that a network administrator is manually inspecting the spoofed ICMP Echo replies, the administrator can visit the information page on our scanning server for details about the experiment and to opt out of future scans, as indicated previously. Three ASs requested to be excluded from future scans.

Disclosure. We disclosed our results to CERT/CC, directly contacted the domains' owners in Table 4, and collaborated with the Shadowserver Foundation to notify affected parties. We also notified the national CERT of our country. One affected party has acknowledged the vulnerability and associated it with a flaw in the 6in4 configuration of their home routers. Other affected ASs have attributed the vulnerability to their clients' setups.

The newly discovered vulnerabilities in this paper have been assigned the identifiers CVE-2024-7596 (GRE and GRE6), CVE-2024-7595 (GUE), CVE-2025-23018 (4in6 and IP6IP6) and CVE-2025-23019 (6in4).

4 Experimental Results

In this section, we analyse our scan results, try to determine if others are also aware of the issue, and discuss abuse reports.

4.1 Scanning Results

All combined, we carried out 26 scans, with the results shown in Table 2 and 3. The scans were performed sporadically for a period of one year, beginning from April 2023 to February 2024. In total, we detected 3,527,565 vulnerable IPv4 addresses and 735,628 vulnerable IPv6 addresses. The approximate number of probed addresses for IPv4 and IPv6 is 3.7 billion and 10 million, respectively. Compared to Yannay's standard scan for vulnerable IPIP hosts in 2020, which discovered roughly 150k vulnerable hosts [39, 40], our scanning methods discovered a total of 530,100 vulnerable IPIP hosts, and 4,263,193 vulnerable tunnelling hosts overall. We can also make the following notable observations:

4.1.1 Host overlap. With the 6in4 and 4in6 TTL Expired scans, we collected both IPv4 and IPv6 addresses from the replies. As a result, some of these addresses will likely correspond to the same hosts.

4.1.2 Scanning method analysis. In Table 2 and 3, column *Replied* refers to the number of replies gathered by each scan, and column *Unique* refers to the unique hosts only identified by the specific scan. Each scan method detected vulnerable hosts that were not discovered by any other scanning methods, demonstrating the importance of each scan method.

The 6in4 scan identified the most vulnerable hosts in absolute numbers, namely 2,126,018 hosts, while the largest percentage of vulnerable hosts identified by a single scan was by the 4in6 TTL Expired scan, where roughly 1.27% of scanned hosts were vulnerable. The 6in4 and 4in6 protocols were created to make the transition to IPv6 easier and allow communication between the two versions of the IP protocol [20]. These protocols may be more widely adopted due to the slow adoption of native IPv6 networks, thereby possibly resulting in more vulnerable hosts.

While testing our scanning module, we also noticed that some hosting providers drop outgoing IPv4 ICMP TTL Expired packets. It is unclear why this was the case.

4.1.3 Geolocation analysis. The vulnerable IPv4 and IPv6 hosts have a diverse geolocation. To gather the locations and ASs of where these hosts reside, we used the *IPtoASN* database, which is an open-source IP location database [30]. We identified 218 territories out of the 249 as having vulnerable hosts. This is illustrated in Figure 3, where almost all territories are coloured. The most significant number of vulnerable hosts belong to the Asia-Pacific Network Information Centre (APNIC), while the region with the least vulnerable hosts is the African Network Information Centre (AFRINIC).

Figure 4a, displays the most prevalent vulnerable countries based on the number of vulnerable addresses in each territory. We use the term *prevalent* to define countries that appear most in the three categories, i.e., number of vulnerable hosts, number of spoofing hosts and number of IPv6 vulnerable hosts. Interestingly, most vulnerable hosts in France have been collected through the 4in6 and 6in4 scans, while Chinese vulnerable hosts are prevalent throughout all IPv4 scans.

The percentages of vulnerable IPv4 hosts that are in China, France, Japan, the USA, Brazil, Australia and India are approximately 0.52%, 0.56%, 0.08%, 0.01%, 0.16%, 0.33%, 0.33%, respectively. The top 5 most vulnerable countries concerning the percentage of vulnerable IPv4 hosts are Benin,

Table 2: Scan results for IPv4 and IPv6 hosts. The total vulnerable hosts are shown followed by the results of each scan type.

		Standard Scan		Subnet Spoof Scan		Spoofing Scan		Echo/Reply Scan		TTL Expired Scan	
Protocol	Total	Replied	Unique	Replied	Unique	Replied	Unique	Replied	Unique	Replied	Unique
IPIP	530,100	88,123	8282	182,668	36,180	66,288	2068	411,565	256,225	105,833	21,820
GRE	1,548,251	612,479	159,673	517,331	394,932	219,213	219,213	509,433	95,735	193,629	145,641
IP6IP6	217,641	860	319	846	846	333	333	216,080	208,894	224	16
GRE6	1806	286	3	900	366	360	4	1219	238	167	20

Table 3: Mixed IPv4/6 scans. The total number of vulnerable hosts is shown followed by the results of each individual scan.

Proto.	Total	Scan Type	Replied	Unique
4in6	130,217	Spoofing	4113	1158
		TTL Expired	127,810	122,531
6in4	2,126,018	Spoofing	1,650,846	464,155
		TTL Expired	465,304	405,252
		IPv4-Mapped src	664,532	22,498
		6to4 src	1,048,559	85,178

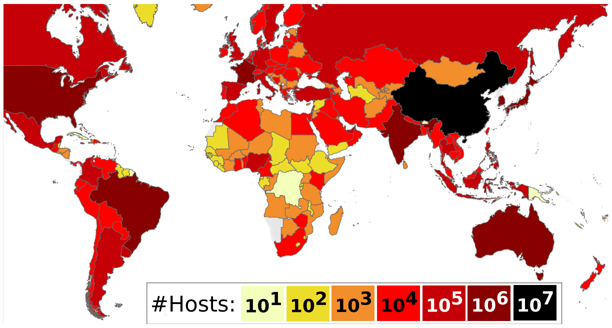


Figure 3: The number of vulnerable IPv4/6 hosts per territory, where colours represent numbers from 10 to 10 million.

Zimbabwe, Jamaica, Laos and Monaco with 2.78%, 2.69%, 1.45%, 1.4% and 1.15%, respectively (see Appendix).

4.1.4 Spoofing-capable hosts. A total of 4276 ASs from 173 countries contained hosts that were able to completely spoof their source IP address, i.e., contained spoofing-capable hosts (not including subnet-spoofing hosts). Over all scans combined, we detected 1,858,892 spoofing-capable hosts. The red bars (second) in Figure 4a show the number of spoofing-capable hosts in each of the most prevalent countries. Notable is that China has more spoofing-capable hosts than all other countries combined, i.e., China contains approximately 59% of all spoofing-capable hosts as identified by our scans. The top 5 countries with the most spoofing-capable hosts are China, Australia, Brazil, India and the USA. In terms of ASs, the red bars (second) in Figure 4b show the number of spoofing-capable hosts in each of the most prevalent countries. It is clear from this Figure that many spoofing-capable addresses are allocated by the same ASs, with AS Chinanet (CHN) having approximately 24% of all identified spoofing addresses. Furthermore, all top 5 most spoofing-capable ASs belong to the APNIC.

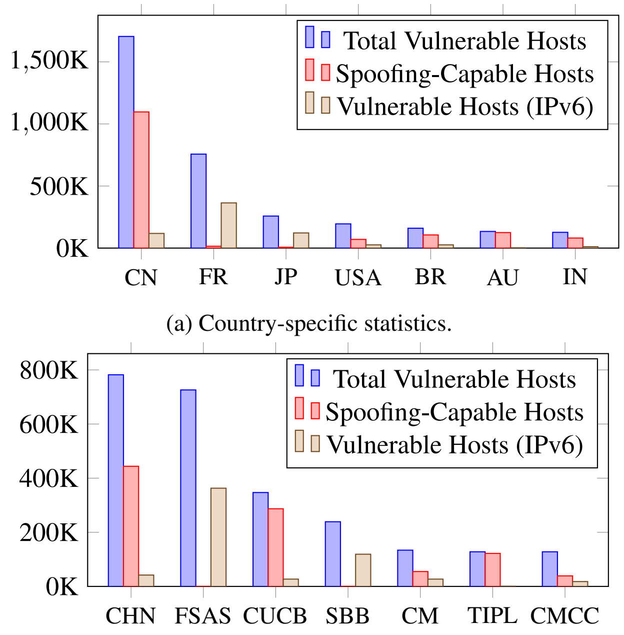
4.1.5 Autonomous System analysis. As of the writing of this paper, there are over 110k allocated ASs [4]. We have identified a total of 11,027 vulnerable ASs, indicating that roughly 1% of ASs had at least one vulnerable host. Even though the number of vulnerable hosts is widely distributed between ASs, it is not evenly distributed amongst them. Figure 4b shows that certain ASs are exceptionally vulnerable. The most vulnerable ASs are CHN, Free SAS (FSAS), China Unicom China169 Backbone (CUCB), Softbank BB Corp. (SBB), China Mobile (CM), TPG Internet Pty Ltd (TIPL), China Mobile Communications Corporation (CMCC). ASs CHN and FSAS have roughly 35% of vulnerable hosts among themselves, while in terms of IPv6, AS FSAS has 49% of vulnerable hosts. What is also evident is the re-occurrence of particular ASs among the three categories. More specifically, ASs CHN, CUCB and TIPL are prevalent throughout all three categories, while AS CHN is always in the top three. Based on these observations, we conjecture that specific software or configuration practices used by some ASs may be the reason why they are more vulnerable than others.

4.1.6 IPv6 scanning bias. Some territories might have a higher IPv6 adoption than others, leading to a bias towards territories with higher adoption. The countries with the greatest IPv6 adoption, based on addresses accessing Google at the time of writing, are France, Germany and India, with 75.45%, 72.68%, and 70.22% IPv6 adoption, respectively [21]. Since France has the greatest IPv6 adoption, it is expected to have more vulnerable IPv6 hosts. This bias towards France is especially noticeable by the yellow bar (second) in Figure 4a, where it is the most vulnerable country regarding IPv6 hosts. Although having the highest IPv6 adoption, the absolute number of vulnerable IPv6 hosts in France remains concerning, especially considering that other territories, such as Germany, appear exceptionally low on the list of vulnerable countries.

4.1.7 Open ports & domains. We used Shodan's InternetDB API [29], an IP lookup database updated weekly, to identify common open ports and domains on vulnerable hosts. We retrieved information on a subset (10k) of hosts for each

Table 4: Top 3 domains and open ports for each protocol for a subset of hosts. The total number of hosts that were queried is shown, followed by the top 3 domains and open ports. The number next to the domains indicates the number of vulnerable hosts that belong to that domain, while the percentage next to a port indicates how many vulnerable hosts had this port open.

Protocol	Total Hosts	Top 3 ports			Top 3 domains		
		Top 1 port	Top 2 port	Top 3 port	Top 1 domain	Top 2 domain	Top 3 domain
IPIP	10,000	HTTP (54%)	HTTPS (53%)	SNMP (23%)	fbcdn (2156)	myqcloud (881)	aiwan4399 (577)
IP6IP6	3099	HTTPS (79%)	HTTP (67%)	SSH (7%)	fbcdn (2015)	whatsapp (357)	infinitum (65)
GRE	10,000	BGP (37%)	GTP-C (18%)	GTP-U (18%)	telenet (310)	windstream (221)	163data (194)
GRE6	24	HTTP (42%)	SSH (29%)	BGP (25%)	N/A	N/A	N/A
6in4	10,000	NTP (51%)	SNMP (46%)	HTTP (5%)	proxad (620)	49-tataidc (476)	14-tadaidc (327)
4in6	1258	SNMP (43%)	NTP (16%)	HTTP (14%)	bbtec (384)	synology (29)	byteark (27)



(a) Country-specific statistics.

(b) AS-specific statistics.

Figure 4: Country and AS statistics. The y-axis represents the number of vulnerable hosts and the x-axis represents the most relevant countries (4a) or ASs (4b). The first bar (blue) denotes the number of vulnerable hosts, the second (red) bar denotes the spoofing-capable hosts, and the third (yellow) bar denotes the number of vulnerable IPv6 hosts.

protocol. For many of the collected vulnerable hosts, no information was available through Shodan's InternetDB; thus, we continuously queried until we had enough information for each protocol. As a result, we assume that the collected data may be biased towards systems that did not block Shodan's scans and/or had available information about them. Furthermore, for some protocols, such as IP6IP6, GRE6 and 4in6, we have collected information on less than 10k results. Table 4 shows each protocol's top 5 domains and open ports.

GRE ports. The three main open port protocols for GRE hosts in the 10k subset are the Border Gateway Protocol (BGP), GPRS Tunnelling Protocol User Plane (GTP-U) and GPRS Tunnelling Protocol Control (GTP-C). BGP may utilise GRE for its *Tunnel Encapsulation attribute*, as defined in RFC 9012 [48]. This attribute allows BGP routers to exchange packets in disconnected paths. On the other hand, GTP-C and GTP-U are used in mobile networks to transport tunnelled packets [61]. Furthermore, GRE can act as an alternative to these protocols, namely, transfer data in mobile networks [19, 61]. Based on the findings for GRE in Table 4, we infer that vulnerable hosts having ports 2123 (GTP-C) and 2152 (GTP-U) open most likely belong to mobile networks, while hosts port 179 (BGP) are BGP routers.

IPIP & IP6IP6 ports. Unlike GRE, the most common open port protocols for IPIP hosts are HTTP, HTTPS, and Simple Network Management Protocol (SNMP). A total of 2521 vulnerable hosts have only port protocols HTTP and HTTPS open, while 4571 vulnerable hosts have at least port protocols HTTP and HTTPS open at the same time. Similar observations are present in the case of IP6IP6 hosts. Based on the observations of open ports, we hypothesise that many IPIP and IP6IP6 vulnerable hosts are mainly servers.

6in4 & 4in6 ports. For 6in4 hosts, the three most common open ports are Network Time Protocol (NTP), SNMP and HTTP. 3715 vulnerable hosts only have the NTP port open, while 3073 hosts only have the SNMP port open. A total of 1273 vulnerable hosts have only SNMP and NTP ports open simultaneously. With 4in6 hosts, the same three port protocols are prevalent, with SNMP being the most common open port. 447 vulnerable hosts only have the SNMP port open, while 153 only have the NTP port open. Because HTTP/HTTPS are not as prevalent (5% and 14%), we hypothesise that most 6in4 and 4in6 vulnerable hosts are routers.

Domains. For IP6IP6 and IPIP, 80% and 59% of vulnerable hosts have hostnames, respectively. In contrast, only 29% of vulnerable GRE hosts have hostname records. The most prevalent domain amongst IPIP and IP6IP6 vulnerable hosts is fbcdn, which is Facebook's Content Delivery Network (CDN), while for GRE and 6in4, the most prevalent domains are Telenet (Telenet ISP) and Proxad (Free ISP), respectively. For IPIP and IP6IP6, 35.81% and 65.5% of vulnerable hosts have at least one hostname with the substring "cdn", respectively, indicating that they may belong to a CDN. For all other protocols, the percentage of hostnames with the substring "cdn" ranges from 0% to 2.5%.

4.1.8 Scanning Limitations. Our scanning methodology might introduce false negatives due to packet loss. For example, a vulnerable host might receive a probe, decapsulate it, and forward it, but the forwarded packet might be lost in transmission. As a result, our scanner will not catalogue it as vulnerable since it has not received a reply.

Furthermore, when scanning from a specific address, there is always the possibility of getting blocklisted, meaning that initial scans might receive more replies than those conducted at a later time. Ideally, each scan would need to be conducted from a new location and address to guarantee that all hosts being scanned will receive the sent probes.

Scanning from multiple distinct geographic locations is essential to get the most accurate scanning results. In our study, we scanned from three locations, though source address filtering mechanisms could still have impacted our results.

Lastly, spoofing scans have an inherent false negative risk, since a host might filter most spoofed addresses but not all. To identify such hosts, one would need to repeat the spoofing scans multiple times with different spoofed inner source addresses. Due to ethical reasons, we did not use this methodology but opted for a single spoofed address instead.

4.2 Are others also scanning?

To determine whether others were aware of the issues we discovered or were scanning for the known IPIP vulnerability, we set up two servers, one in the UK and one in the US, to monitor incoming traffic for tunnelling packets. This measurement was started on 25 August 2023 and was stopped on 4 February 2024. We mainly detected GRE packets with seemingly random content and an empty or 512-byte UDP payload. We conjecture that these packets are generated by hosts infected by the Mirai botnet [42]. A single IPIP scan was detected on 21 September 2023 that had the payload `GET / HTTP/1.1\r\nHost: www\r\n\r\n`. This is the payload used by Yannay's public IPIP scanning tool.[2] Overall, this shows that few researchers or organisations are scanning for vulnerable tunnelling hosts.

5 Attacks

In this section, we demonstrate that vulnerable hosts enable new DoS attacks. Note that spoofing-capable hosts also enable an attacker to spoof a packet's source address, which in turn enables attacks such as DNS spoofing [28], traditional amplification DoS attacks [22, 45], off-path TCP hijacking [47], SYN floods [17], certain Wi-Fi attacks [63], and so on.

5.1 Routing loop DoS

While testing the 6in4 scan, we observed that an IPv6 packet with as destination `::IPV4_ADDRESS_IN_HEX` causes Linux to loop the packet on the tunnelling interface until the hop limit field reaches zero. That is, this packet is sent and transmitted 256 times on the same interface, resulting in a trivial DoS attack. Note that this is different from the observations of Nakibly and Arvo [2] and the warning in RFC 6324 [44]: our server was not vulnerable to existing looping attacks, but only was vulnerable when using an IPv6 address that starts with zeros and is followed by the encoded IPv4 address.

5.2 Abusing abuse reports

We received two abuse reports while performing our scans. The first was in response to a partial test scan where SYN packets were sent to port 80, and SYN/ACK responses were measured. The second was in response to a standard IP6IP6 scan. In both cases, we explained the research to our hosting provider, and we were allowed to continue our scans, though we did not complete or perform other SYN-based scans.

The identified vulnerable hosts enable an adversary to spoof traffic that triggers abuse reports towards a victim. This can lead to an *administrative DoS attack* since receiving a large number of abuse reports may cause the hosting provider to disable the victim's account. In particular, an adversary can spoof malicious traffic that appears to come from the victim towards a host that is known to quickly send abuse reports. For example, the abuse reports we received were in response to traffic where the source address could have been trivially spoofed. This means an adversary can spoof such traffic towards different organisations and, as a result, cause a flood of abuse reports towards the victim. A flood of abuse reports would, at best, cause an administrative overhead for the victim and, at worst, may cause the termination or temporary suspension of the victim's account. When the victim does not reside on the cloud, these abuse reports can cause system administrators to blocklist the victim's IP address, which can lead to the victim being unable to access large parts of the internet.

Although we did not test such an attack in practice due to ethical concerns, our current observations indicate that such an abusing-abuse-reports attack is feasible.

[2] We contacted Yannay, who informed us this was not one of his scans.

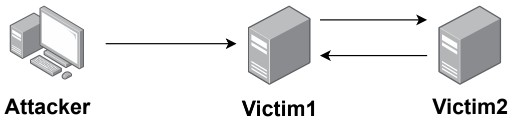


Figure 5: *Ping-Pong Amplification Attack*. The attacker sends the encapsulated packet to the first victim, who will decapsulate the header and forward it to the second victim. This continues until there are no more headers in the datagram.

5.3 Ping-Pong Amplification attack

Our first novel DoS attack, called the *Ping-Pong amplification attack*, loops packets between vulnerable hosts. The idea is that an adversary constructs a tunnelling packet that has another tunnelling packet as an inner packet, and so on, until the maximum packet size, i.e., Maximum Transmission Unit (MTU), is reached. The inner packets' IP header has the other vulnerable host as the destination, meaning the (decapsulated) packet is constantly sent between the two hosts (see Figure 5). For example, given that IP header i has source address A and destination address B, then the next IP header $i+1$ will have source address B and destination address A. The attacker sends the constructed packet to one of the two victims, which will forward the inner packet to the second victim. This continues until there are no inner packets to be decapsulated. Note that the attack is not restricted to only two hosts, i.e., the packet could be constructed to loop between several hosts.

This attack is considered an *amplification attack* because it amplifies the amount of bytes-per-second sent by the attacker. For example, assume that the attacker sends p Ping-Pong packets and that the MTU equals m. Letting a denote the minimum amount of bytes in an IP header, which is 20 for IPv4 and 40 for IPv6 [14, 32], then a single Ping-Pong packet can contain $h = m/a$ headers. Without loss of generality, we will assume that each packet is composed only of IP headers, i.e., that the adversary is abusing the IPIP or IP6IP6 protocol.

When sending a single Ping-Pong packet, the total bytes consumed by the first victim is the sum of all packets received and sent by them. The victim will receive the packet of size m and will forward the packet after stripping a header of a bytes. The second time this packet is received, it will be $m-2a$ bytes since the second victim also stripped a header before forwarding it back. This continues until there are no headers to be stripped. As a result, when sending a single Ping-Pong packet, the total number of bytes θ_1 received and sent by the first host can be calculated as follows:

$$\theta_1 = \sum_{k=0}^{h} (m - a \cdot k) = \frac{1}{2} \cdot (h+1) \cdot (2m - a \cdot h) \approx \frac{h \cdot m}{2} \quad (1)$$

For the second victim, the total bytes θ_2 are the same as θ_1 with the only exception of the initial $k = 0$, which is only being received by the first victim of the attack for every packet:

$$\theta_2 = \theta_1 - m \approx \frac{h \cdot m}{2} - m \approx \frac{h \cdot m}{2} \quad (2)$$

The amplification factor f of the attack equals the total bytes consumed by a victim over the initial total bytes sent by the attacker:

$$f = \frac{\theta_1}{m} \approx \frac{\theta_2}{m} = \frac{h}{2} \quad (3)$$

We can further split θ into the inwards consumed traffic θ_{in} and outwards traffic θ_{out}, where $\theta_{in} \approx \theta_{out} \approx \theta/2$, with the exact calculation for θ_{out} shown in the Appendix.

The theoretical maximum number of IP headers, i.e., inner packets, that can be encapsulated in a single packet is 3276 for IPv4 and 1638 for IPv6, corresponding to a packet of 65,520 bytes [46]. In practice, such a large packet will likely exceed the MTU of the path towards the victim(s) and would require fragmentation at the IP level. However, IPv6 does not support fragmentation [14], forcing the attacker to construct packets that are smaller or equal to the MTU, which in practice typically ranges between 1280 and 1500 bytes [25].

As an example, when constructing an IPIP packet of 1500 bytes, it can contain 75 headers. The total traffic caused by this packet to the first victim is 56,250 bytes, resulting in an amplification factor of 37.5. When sending an IPIP packet of 3000 bytes instead, the amplification factor would be 75. Furthermore, the theoretical maximum amplification factor for IPv4 and IPv6 would be 1638 and 819, respectively. This theoretical maximum would rely on all in-between nodes having an MTU of at least 65,520. Against IPv4, the use of IP fragmentation may alternatively be exploited.

5.3.1 Economic Denial Of Sustainability (EDoS)

EDoS refers to elevating a victim's costs on the cloud [56]. These attacks are possible since many cloud providers use a *self-scaling* model, automatically adapting resources to ensure availability and protect a host from a DoS attack [55]. Thankfully, some cloud providers only charge their clients based on outgoing traffic, obsoleting traditional EDoS attacks that overload a victim with inbound traffic to elevate costs.

Unfortunately, our *Ping-Pong* attack causes a host to generate large amounts of outbound traffic, making an EDoS attack possible. A vulnerable host residing in the cloud could be forced to forward traffic, counting towards outbound traffic and causing additional costs to the victim.

5.4 Tunnelled-Temporal Lensing (TuTL)

In this DoS attack, the adversary sends traffic over multiple different chains of vulnerable hosts, such that this traffic arrives simultaneously at the victim. In other words, it abuses tunnelling to create a TuTL effect on the victim. The core

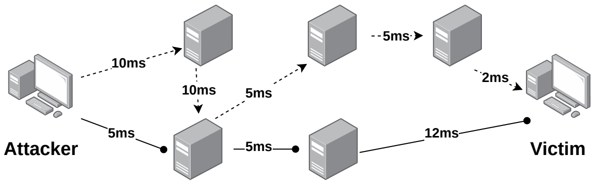


Figure 6: Example TuTL attack. The attacker sends two packets across two different paths (dotted line and solid line), with each edge denoting the latency of that connection.

idea is that we can recursively encapsulate tunnelling packets within each other to create different paths, i.e., chains of hosts a packet will go through before reaching the victim. Each path takes a different amount of time for the most-inner packet to reach the victim. By carefully scheduling when to send packets over different paths, all most-inner packets will arrive simultaneously at the victim.

For example, assume the attacker identified five vulnerable hosts as shown in Figure 6. The attacker then estimates all latencies between the hosts, attacker, and victim. Once the latencies are known, different paths are constructed towards the victim. In Figure 6, two example paths are shown: the dotted line path with a total latency of 32ms and the solid line path with a total latency of 22ms. The attacker then sends a burst of packets over the dotted line path and, after 10ms, sends a second burst of packets over the solid line path. Consequently, the two packet bursts will arrive at the victim simultaneously, resulting in a peak of inbound traffic called the *pulsing window*. This pulse of inbound traffic is similar to pulsing and shrew attacks and causes benign traffic to be dropped [36,41].

Our TuTL attack consists of three phases: the *latency collection*, *path exploration*, and *path and traffic scheduling*.

5.4.1 Latency Collection

Given a list of vulnerable hosts, the following three types of latencies are first collected:

1. Attacker to vulnerable host: To estimate the latency α between the attacker M and a vulnerable host A, the attacker will send a tunnelled packet to the vulnerable host, which will decapsulate it and return the inner packet back to the adversary. The path followed by the probe will be $M \rightarrow A \rightarrow M$. The attacker can then estimate the packet's Round-Trip Time (RTT) and calculate the latency α as follows:

$$\alpha \approx \frac{RTT(M \rightarrow A \rightarrow M)}{2} \tag{4}$$

This assumes that the latency between the attacker and hosts is equal in both directions.

2. Vulnerable host to vulnerable host: To estimate the latency β between two vulnerable hosts A and B, the attacker M sends a probe over the path $M \rightarrow A \rightarrow B \rightarrow M$, i.e., the probe passes through both hosts and then returns to the attacker. Let α^* denote the latency of $B \rightarrow M$, i.e., between the attacker and the second host, calculated similarly as in Equation 4. The latency β between A and B can then be estimated as:

$$\beta = RTT(M \rightarrow A \rightarrow B \rightarrow M) - \alpha - \alpha^* \tag{5}$$

Here α comes from Equation 4. Unlike previously, we assume that the latency from A to B may differ from B to A; thus, the attacker must estimate both latencies for increased accuracy.

3. Vulnerable host to victim: To estimate the latency c between a host A and a victim V, the attacker first estimates the latency v between itself and the victim V, i.e., of the path $V \rightarrow M$. To achieve this, we assume that the victim runs a public service that replies to external requests, e.g., a web service that replies with a SYN/ACK in response to a TCP SYN. Similar to Equation 4, we can then send probes and estimate the latency v. The attacker then sends a tunnelled probe over the path $M \rightarrow A \rightarrow V \rightarrow M$, i.e., the probe reaches the host, the victim, and then the victim replies with a SYN/ACK. Here the probe's inner-most header must have a spoofed source address, i.e., the attacker's address, so that the SYN/ACK is sent back to the attacker. The latency c between a vulnerable host and the victim can then be estimated by:

$$c = RTT(M \rightarrow A \rightarrow V \rightarrow M) - \alpha - v \tag{6}$$

Here α comes from Equation 4.

To eliminate outliers, we collect 1000 RTTs for each latency and pick the median as the latency estimate.

5.4.2 Path Exploration Algorithm

The next step in our attack is to construct paths with different latencies towards the victim. To simplify this task, we first gather paths of increasingly large lengths d, where the length of a path is defined as the number of vulnerable hosts a packet travels through. To calculate the latency ℓ of a path, we first define α as the latency between the attacker and the first host. Furthermore, let β_i be the latency between the host in position i and $i+1$ of the path, and let c be the latency between the last host of the path and the victim. We get:

$$\ell = \alpha + \sum_{i=1}^{d-1} (\beta_i) + c \tag{7}$$

We created a Breadth-First-Search (BFS) method to identify possible paths towards the victim. This BFS search prioritises paths of lower depths. The BFS algorithm calls a function which adds hosts to a path until the given depth is reached. Before adding a path to the solutions, it ensures that the current path introduces a variance in latency compared to existing paths to avoid choosing paths with latencies that are too similar. This latency variance threshold is set to 10ms.

The function also tracks the number of bytes each host will process under the selected paths. If a host exceeds a set amount of bytes, no new paths can include that host. This ensures no intermediate host suffers from a DoS, which would otherwise delay packets' arrival at the victim. Moreover, at each depth of the function, we randomise the list of available hops to distribute the traffic each host will receive.

The limit on the number of bytes that a host can process also serves as an implicit stop condition for the BFS: when there is an insufficient number of hosts available to cover a given depth, the search can terminate.

5.4.3 Path & Traffic Scheduling

After identifying paths, the attacker sorts them based on latencies and estimates the switch time st_i between paths. A switch time is when the attacker must switch from sending bursts of packets over path p_i to sending packets over the next path p_{i+1}. For instance, in Figure 6, the attacker sends bursts of packets over the path with a latency of 32ms, and after 10ms, switches to sending packets over the path with 22ms latency. This ensures the bursts arrive almost simultaneously at the victim. In general, the switch time st_i equals:

$$\forall i \in \mathbb{Z}, 0 \leq i < N: \, st_i = \ell_i - \ell_{i+1} \tag{8}$$

where N is the number of paths and ℓ_i and ℓ_{i+1} are the latencies of the current and following path.

With the switch times determined, the attack can start by sending bursts of packets over the path with the biggest latency, and when the switch time for that path has expired, sending bursts of packets over the next path, etc. We call the time frame where bursts of packets are sent to a specific path the *sending window*. The burst size should be big enough to generate enough traffic towards the victim, but small enough to ensure the attacker does not miss the next sending window. We can estimate the resulting amplification factor a of our TuTL attack by

$$a = \frac{b_v}{b_a} \tag{9}$$

where b_a is the number of packets that the attacker sends during a given duration, i.e., the attacker's bandwidth, and b_v is the number of packets the victim receives over a chosen 20ms window, i.e., the victim's bandwidth at specific 20ms. Estimating the victim's bandwidth this way prevents the amplification factor from being influenced by outliers. For example, if a victim receives 95% of the traffic in 20ms but will then receive the other 5% over 500ms, the amplification factor would have been severely influenced when measuring the victim's bandwidth over the whole receive duration.

5.4.4 Experiments & Results

Experimental setup. To evaluate the TuTL attack in practice, we cannot use real vulnerable hosts due to ethical concerns, and instead, we set up five vulnerable IP6IP6 hosts. Two servers are in the US, one in France and two in Singapore. The diverse locations of these servers allow us to collect latencies simulating real/non-trivial networks in the wild. We restricted the size of each packet sent by the attacker to 1280 bytes (32 headers) to ensure that an intermediary router would not drop any packet. Additionally, we significantly reduce the paths towards each vulnerable host by ensuring they will receive no more than 200Mb/s of inwards or outwards traffic during the attack simulation. We conducted several experiments with different amounts of hosts to analyse how the attack scales. Furthermore, we ran each experiment ten times and estimated the average amplification factor and the standard deviation.

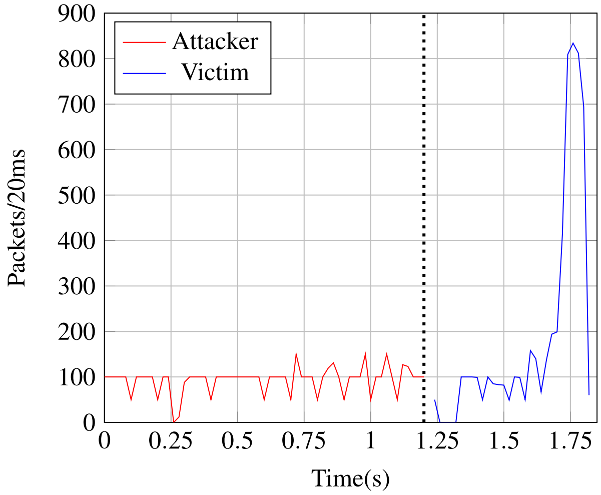


Figure 7: TuTL experiment with two hosts (5700 packets, 16 paths, sent in 1.20s, received in 0.58s). The X-axis shows, on the left of the dotted horizontal line, the number of packets sent by the attacker, and on the right the number of packets received by the victim, in buckets of 20ms. Time on the X-axis is synchronised between victim time and attacker time.

To measure the amplification factor of the TuTL attack, we use Equation 9. The enumerator will consist of the packets received by the victim at the tallest 20ms bucket over 20ms, while the denominator will consist of the packets sent by the attacker over the whole sending period. We do this to estimate the worst-case scenario a victim will have to defend against.

Resulting amplification factors. An example of our experiments with two vulnerable hosts can be seen in Figure 7. The time is synchronised between the victim and the attacker. In the experiment, the pulse is noticeable as a peak that the victim receives in a 20ms bucket (examples with 3, 4 and 5 vulnerable hosts are in the Appendix). When we use two, three, four, and five hosts, the average amplification factor is 11, 13, 14, and 16, respectively, while the standard deviation is 2, 2.5, 2, and 2.6, respectively. We can deduce that the attacker, with as few as two hosts, can map their traffic at a shorter time frame and thereby amplify their bandwidth. With more hosts, the attacker's amplification factor increases, as seen by the taller peak at the victim.

DFS vs BFS. In Figure 7, where only two hosts are introduced, we obtain the longest attack time of 1.20 seconds. The attack time is longer than all other experiments since, with only two hosts the BFS acts more like a Depth-First-Search (DFS). For example, assume attacker M, hosts A and B and victim V. The path $M \rightarrow A \rightarrow B \rightarrow V$ will have the same latency with all other paths of depth 2, namely $M \rightarrow B \rightarrow A \rightarrow V$. The same holds with all other depths greater than 1. In contrast, this does not hold when introducing more hosts. For example, path $M \rightarrow A \rightarrow C \rightarrow V$ is unlikely to have the same latency as path $M \rightarrow A \rightarrow B \rightarrow V$. As a result, when only two hosts are present, and since we only choose one path from each depth because of the 10ms difference threshold, we will retrieve longer paths with longer total latencies.

Latency differences between paths. In Figure 7, it is also apparent that some of the packets miss their pulsing window at the victim and arrive earlier than expected. All experiments present this behaviour when acquiring paths with a DFS-like approach. This is because, with a DFS-like approach, the paths have bigger latency differences (approximately 100ms difference). When this phenomenon occurs, multiple bursts are sent before switching to the following path (seen by the red peaks in Figure 7), exceeding the 20ms buckets we analyse. Finally, with this observation, we can deduce that latency differences between paths greater than 20ms will cause some packets to miss their pulse window. In contrast, having a very small latency difference between each path will cause the attacker to miss the next sending window, making some packets arrive later. When more vulnerable hosts are introduced, lower latency paths will be added first, meaning latencies closer to each other but still respecting the 10ms threshold difference will be chosen first. Since we introduce more hosts at each experiment, more paths with closer latencies are selected; thus, more packets will arrive at the same 20ms bucket.

5.4.5 Limitations and Discussion

MTU. Respecting an *MTU* in each packet means an attack's theoretical maximum latency is significantly reduced. The typical *MTU* for an interface is approximately 1280 to 1500 bytes [25], corresponding to 75 IP headers in IPIP or 37 IPv6 headers for IP6IP6. Furthermore, an adversary must ensure that each vulnerable host's *MTU* is great enough to support the attacker's traffic, especially for early receiving hosts in long paths.

Furthermore, since with IPv6, intermediary routers will drop packets exceeding their *MTU* [14], an attacker using a tunnelling protocol such as IP6IP6 must ensure that the packet being sent is smaller in size than the respective *MTU* of each node in the path to the destination. Furthermore, fragmentation can introduce latency differences concerning the attacker's original anticipated path latency due to the reassembly of the fragmented packets at the victim's side [33].

Host Bandwidth. Another attack limitation stems from the attacker's need to ensure that the intermediary hosts' bandwidth is not exceeded. When a host's bandwidth is exceeded, packets might be dropped or significantly delayed, impacting estimated latencies and the resulting attack efficiency.

Latency. Measuring the latency with our methodology becomes increasingly more time-consuming as the number of vulnerable hosts available for the attack increases. Additionally, since the sent packets will be encapsulated and checked before being forwarded back, these computations can also increase the variance of the RTT for longer paths. That being said, this was not a noticeable issue with the vulnerable tunnelling hosts used in our experiments.

A second aspect to consider is the latency's evolution over short and longer durations. If latencies change significantly over short periods, such as during a lensing attack, the attack's efficiency decreases. If the latencies change over longer periods, the adversary would have to measure them before each attack. However, in the original lensing attack of Rasti et al., the short-term variation in latencies towards DNS servers was measured, and it was found that most servers do not suffer from widely varying latencies [50, §3]. Additionally, in their experiments, they found that path latencies exhibited very little variance over a longer duration of at least 10 days. We conjecture that the same is true for most vulnerable tunnelling hosts. To validate this, we conducted the TuTL attack with 5 servers with three-day-old latencies. We gathered an average amplification factor of 12.6 and a standard deviation of 1.98. It is then apparent that the latencies do not change significantly in a few days to render the TuTL attack obsolete to conduct, even though the difference in amplification is observable.

Reduced Packet Sizes at Victim. When using paths with a large depth, more tunnelling packets need to be recursively encapsulated within each other, while only the most-inner packet arrives at the victim. With long paths, the packet arriving at the victim can be substantially smaller than the original packet sent by the attacker. However, this problem is reduced by using a BFS algorithm to collect paths since it prioritises paths of shorter depth. Moreover, long paths are only needed when fewer vulnerable tunnelling hosts are available. Additionally, even when longer paths are used, we can assume that amplification in terms of the number of packets is still severe since for each packet a victim receives, they must waste computational resources to process and validate the packet [9].

6 Defences

In this section, we will discuss two types of defence groups, namely *Host Defences* and *Network Defences*.

6.1 Host Defences

Host Defences can be deployed by hosts without relying on the network for security. A first defence that can be employed is to only accept tunnelling packets from trusted source IP addresses. This would cause a host to stay hidden from Internet-wide scans and prevent it from accepting tunnelling packets from arbitrary sources. An IP address to which the host previously transmitted packets could be an example of a trusted source IP address. The host could maintain a list of IPs for which it will decapsulate packets based on previous interactions, and tunnelled packets from different IPs should be dropped. Unfortunately, since the discussed tunnelling protocols do not use authentication, an adversary that can spoof source IP addresses can still send tunnelling packets to the host by spoofing a trusted source address.

In the case of mGRE, or any multipoint tunnelling setup where the remote address must be left empty, the host should ensure that incoming traffic is only served if it originates from desired addresses or known neighbours. This can be done by configuring the firewall of that device to drop encapsulated packets where the source address is from a non-trusted source.

As mentioned in Section 2, vulnerable hosts implementing GRE tunnels can remain hidden from an attacker's Internet-wide scan by utilising a non-default GRE key. Unfortunately, this key field is not an ideal authentication mechanism, and if explicitly targeted, an attacker can trivially learn the key from unencrypted packets or even brute-force the key.

Ideally, a host should use a more secure set of protocols to provide authentication and encryption, e.g., IPsec [18]. Since IPsec can transport any IP protocol, it can be used to protect all discussed tunnelling protocols: a host should only accept tunnelling packets that are protected using IPsec.

6.2 Network Defences

Network Defences can be implemented on routers or other Internet middle-boxes to prevent or limit the damage of attacks. First, as shown in Table 2 and 3, many discovered hosts can completely spoof their source address due to inadequate source address filtering. Consequently, a network can prevent an adversary from forcing hosts to spoof packets by incorporating ingress and egress traffic filtering [54].

Deep packet inspection can also be used to detect likely malicious tunnelling packets. For instance, several of our attacks involve sending recursively nested tunnelling packets. Such unusual packets can be detected and dropped to limit the damage that an attacker can inflict. For example, the network could drop packets where the number of encapsulated headers exceeds a number x, where x is the number of tunnelled hosts in the network. Deep packet inspection could then be expanded to inspect other characteristics indicating that an encapsulated packet might have malicious intent. For example, inner packets with TTL value 0 should be dropped since such a packet cannot exist in the wild and can be used by attackers to identify vulnerable hosts. To the best of our knowledge, this is a newly suggested defence.

In some networks, it may also be possible to block all ingoing or outgoing unencrypted tunnelling packets. For instance, if a host uses IPsec in combination with GRE but, due to a misconfiguration, also accepts unencrypted GRE packets, the network can block unencrypted GRE packets. The host would still be able to receive IPsec traffic and hence function normally while being protected from attacks.

7 Related Work

Tunnelling security. Closely related work is an excellent 1-page report by Yannay where the IPv4 Internet is scanned for IPIP hosts using a standard scan [39]. This inspired us to design additional scanning methods, scan the IPv6 Internet, test other tunnelling protocols, including GRE, GUE, 4in6, and 6in4, and investigate DoS attacks. While Yannay discovered roughly 150k vulnerable IPIP hosts [40], we discovered 530,100 vulnerable IPIP hosts by incorporating additional scanning methodologies. We also scanned for other IPv4/6 tunnelling protocols, and we explored DoS attacks. Phenoelit discussed how to inject traffic into a GRE tunnel but did not consider GRE hosts that accept traffic from any source [38,64]. The security of IPv6-in-IPv4 has also been studied in various works [1,3,11,23,34]. Kristoff et al. conducted an Internet-wide scan searching for open 6in4 hosts revealing 1,546,843 vulnerable IPv4 hosts [34], though their methodology only consisted of the ICMPv6 Echo/Reply scan.

Temporal Lensing. Rasti et al. introduced temporal lensing to perform DoS attacks [50]. They abused DNS recursion to create traffic pulses and used the Kings method of estimating latencies by utilising DNS resolvers close to the DNS victim [24]. These DNS resolvers are utilised to cause a temporal lensing attack through a scheduler optimised to choose which DNS resolver based on their latency and the probability that the message will arrive at a specific pulsing window.

Rasti et al. experimented with public DNS resolvers, resulting in an estimated amplification factor of 14 with a low-bandwidth attacker (500pps and 75 hosts) and 5 with a high-bandwidth attacker (20,000pps and 1201 hosts) [50]. It is non-trivial to compare this to our results because, due to ethical reasons, we were limited in using our own vulnerable tunnelling hosts during the evaluation, while Rasti et al. could use 3000 public DNS servers. Nevertheless, using only 5 vulnerable tunnelling hosts, we already achieved an average amplification factor of 16, and we conjecture that using more vulnerable hosts would further increase the amplification factor. One advantage of our TuTL attack is that longer paths can be created, resulting in a higher amplification factor.

Spoofing-capable networks. Beverly et al. identify networks from which the client can spoof their source address, and they measure how many IP addresses are spoofable [6]. Their methodology allows clients to test their networks using a "spoofer" software that will send spoofed UDP traffic to a researcher-controlled server. The project has since expanded to include IPv6 addresses [7]. The latest results, as of the writing of this paper, show that 560 IPv4 (including NAT) and 252 IPv6 ASs can at least partially spoof.

Kührer et al. use broken public DNS proxies to identify ASs that allow spoofing [35]. A DNS proxy is considered broken if it forwards the DNS request without first changing the source IP, ultimately forwarding packets which should be filtered for spoofing. By sending DNS queries to these broken DNS proxies and receiving the traffic back, they catalogue the AS where the DNS proxy resides as poorly filtered. The paper identified a total of 2692 ASs that allow spoofing.

While the two aforementioned papers have discovered a total of 560 and 2692 spoofing ASs, respectively, our paper showcases that a total of 4276 ASs have the ability to spoof.

8 Conclusion

This paper is the first to systematically analyse the security of tunnelling hosts on the IPv4 and IPv6 Internet. Our large-scale Internet-wide scans identified over 4 million hosts that accept unencrypted tunnelling packets from any source. This is concerning because vulnerable hosts can be abused as one-way proxies, and many of these hosts also allow an adversary to spoof a packet's source address, enabling various kinds of known and novel attacks. Moreover, we also demonstrated that these vulnerable hosts enable novel DoS attacks, such as our *TuTL* and *Ping-Pong* attacks. The TuTL attack is especially concerning since it can be abused to perform DoS attacks against any third-party host on the Internet. Our measurements also show that many Autonomous Systems, more than four thousand in total, do not (properly) implement source address filtering, thereby allowing the spoofing of source IP addresses. We hope our results will motivate and guide administrators to secure tunnelling hosts better.

Acknowledgments

This research is partially funded by the Research Fund KU Leuven and the Cybersecurity Research Programme Flanders.

Open science

We provide: (1) the used ZMap version along with our scanning modules to identify vulnerable tunnelling hosts of different protocols (Section 4.1); (2) the attack scripts for the Tunnelling-Temporal Lensing (TuTL) attack (Section 5.4); and (3) the scripts which allow system administrators to identify whether hosts in their networks may be vulnerable to any of the identified vulnerabilities (Section 3.2).

The artifacts can be found at https://doi.org/10.5281/zenodo.15548968 and at https://github.com/vanhoefm/tunneltester.

References

[1] Shubair A Abdulla. Survey of security issues in IPv4 to IPv6 tunnel transition mechanisms. *International Journal of Security and Networks*, 12(2):83–102, 2017.

[2] Gabi Nakibly Michael Arov. Routing loop attacks using IPv6 tunnels. In *USENIX WOOT*, 2009.

[3] Zeeshan Ashraf, Adnan Sohail, Sohaib A Latif, Abdul Hameed Pitafi, and Muhammad Yousaf Malik. Challenges and mitigation strategies for transition from IPv4 network to virtualized next-generation IPv6 network. *Int. Arab J. Inf. Technol.*, 20(1):78–91, 2023.

[4] Number resource allocation data. https://www.iana.org/numbers/allocations/, 2024. Accessed on March 12, 2024.

[5] Fred Baker and Pekka Savola. Ingress Filtering for Multihomed Networks. RFC 3704, March 2004.

[6] Robert Beverly and Steven Bauer. The spoofer project: Inferring the extent of source address filtering on the internet. In *USENIX SRUTI*, volume 5, pages 53–59, 2005.

[7] Robert Beverly and Steven Bauer. State of IP spoofing, 2023.

[8] Brian E. Carpenter and Keith Moore. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056, February 2001.

[9] Danilo Cerović, Valentin Del Piccolo, Ahmed Amamou, Kamel Haddadou, and Guy Pujolle. Fast packet processing: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):3645–3676, 2018.

[10] Liting Chang, Keyu Lu, Chao Li, and Zhaoxin Zhang. ZMap performance in open DNS resolver discovery. In *ACCTCS*, 2022.

[11] Lorenzo Colitti, Giuseppe Di Battista, and Maurizio Patrignani. Discovering IPv6-in-IPv4 tunnels in the internet. In *IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2004.

[12] Dr. Steve E. Deering and Alex Conta. Generic Packet Tunneling in IPv6 Specification. RFC 2473, December 1998.

[13] Dr. Steve E. Deering and Bob Hinden. IP Version 6 Addressing Architecture. RFC 4291, February 2006.

[14] Dr. Steve E. Deering and Bob Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, July 2017.

[15] Gopal Dommety. Key and Sequence Number Extensions to GRE. RFC 2890, September 2000.

[16] Zakir Durumeric, Michael Bailey, and J Alex Halderman. An Internet-Wide view of Internet-Wide scanning. In *USENIX Security*, 2014.

[17] Wesley Eddy. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987, August 2007.

[18] Sheila Frankel and Suresh Krishnan. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, February 2011.

[19] Jens Gebert and Andreas Wich. Comparison of provider backbone bridging, TRILL, GRE and GTP-U in 5G for time sensitive industrial applications. In *2018 IEEE CSCN*, pages 1–6, 2018.

[20] Robert E. Gilligan and Erik Nordmark. Basic Transition Mechanisms for IPv6 Hosts and Routers. RFC 4213, October 2005.

[21] Per-country IPv6 adoption, 2024. Accessed on February 6, 2024 from https://www.google.com/intl/en/ipv6/statistics.html#tab=per-country-ipv6-adoption.

[22] Harm Griffioen, Kris Oosthoek, Paul van der Knaap, and Christian Doerr. Scan, test, execute: Adversarial tactics in amplification DDoS attacks. In *ACM CCS*, 2021.

[23] Kejun Gu, Liancheng Zhang, Zhenxing Wang, and Yazhou Kong. Comparative studies of IPv6 tunnel security. In *ICNC-FSKD*. IEEE, 2017.

[24] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: estimating latency between arbitrary internet end hosts. In *Workshop on Internet Measurment*. ACM, 2002.

[25] Matthias Göhring, Haya Shulman, and Michael Waidner. Path MTU discovery considered harmful. In *ICDCS*, 2018.

[26] Tom Herbert. UDP encapsulation in linux. In *The Technical Conference on Linux Networking*, 2015.

[27] Tom Herbert, Lucy Yong, and Osama Zia. Generic UDP Encapsulation. Internet-Draft draft-ietf-intarea-gue-09, Internet Engineering Task Force, October 2019. Work in Progress.

[28] Amir Herzberg and Haya Shulman. Fragmentation considered poisonous, or: One-domain-to-rule-them-all.org. In *2013 IEEE Conference on Communications and Network Security (CNS)*, pages 224–232. IEEE, 2013.

[29] InternetDB. https://internetdb.shodan.io/. Accessed on May 17, 2024.

[30] IP to ASN. https://iptoasn.com/, 2024. Accessed on February 6, 2024.

[31] *ip-tunnel(8) Linux User's Manual*, December 2011.

[32] Internet Protocol. RFC 791, September 1981.

[33] Christopher A. Kent and Jeffrey C. Mogul. Fragmentation considered harmful. *SIGCOMM Comput. Commun. Rev.*, 25(1):75–87, jan 1995.

[34] John Kristoff, Mohammad Ghasemisharif, Chris Kanich, and Jason Polakis. Plight at the end of the tunnel: Legacy IPv6 transition mechanisms in the wild. In *PAM 2021*, Berlin, Heidelberg, 2021. Springer-Verlag.

[35] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Exit from hell? reducing the impact of Amplification DDoS attacks. In *USENIX Security*, San Diego, CA, 2014.

[36] Aleksandar Kuzmanovic and Edward W Knightly. Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In *ACM SIGCOMM*, 2003.

[37] Tony Li, Dino Farinacci, Stanley P. Hanks, David Meyer, and Paul S. Traina. Generic Routing Encapsulation (GRE). RFC 2784, March 2000.

[38] Felix Lindner. GRE: Attacking generic routing encapsulation. Accessed on February 4, 2024 from https://web.archive.org/web/20200216140042/http://www.phenoelit.org/irpas/gre.html, 2000.

[39] Yannay Livneh. Spoofing IP with IPIP. *Proof of Concept or GTFO*, 21:7, 2022.

[40] Yannay Livneh. Modern adventures with legacy protocols. *Insomni'Hack*, 2023.

[41] Xiapu Luo, Rocky KC Chang, et al. On a new class of pulsing denial-of-service attacks and the defense. In *NDSS*, 2005.

[42] Joel Margolis, Tae Tom Oh, Suyash Jadhav, Young Ho Kim, and Jeong Neyo Kim. An in-depth analysis of the mirai botnet. In *ICSSA*, pages 6–12. IEEE, 2017.

[43] Austin Murdock, Frank Li, Paul Bramsen, Zakir Durumeric, and Vern Paxson. Target generation for Internet-Wide IPv6 scanning. In *IMC*, IMC '17. Association for Computing Machinery, 2017.

[44] Gabi Nakibly and Fred Templin. Routing Loop Attack Using IPv6 Automatic Tunnels: Problem Statement and Proposed Mitigations. RFC 6324, August 2011.

[45] M. Nawrocki, J. Kristoff, R. Hiesgen, C. Kanich, T. C. Schmidt, and M. Wählisch. SoK: A data-driven view on methods to detect reflective amplification DDoS attacks using honeypots. In *EuroS&P*. IEEE, 2023.

[46] Tammy Noergaard. Chapter 4 - the fundamentals in understanding networking middleware. In Tammy Noergaard, editor, *Demystifying Embedded Systems Middleware*, pages 93–190. Newnes, Burlington, 2010.

[47] Yepeng Pan and Christian Rossow. TCP spoofing: Reliable payload transmission past the spoofed TCP handshake. In *IEEE S&P*. IEEE Computer Society, 2024.

[48] Keyur Patel, Gunter Van de Velde, Srihari R. Sangli, and John Scudder. The BGP Tunnel Encapsulation Attribute. RFC 9012, April 2021.

[49] Charles E. Perkins. IP Encapsulation within IP. RFC 2003, October 1996.

[50] Ryan Rasti, Mukul Murthy, Nicholas Weaver, and Vern Paxson. Temporal Lensing and Its Application in Pulsing Denial-of-Service Attacks. In *IEEE S&P*, 2015.

[51] Internet Control Message Protocol. RFC 792, September 1981.

[52] T. Saad, B. Alawieh, H. T. Mouftah, and S. Gulder. Tunneling techniques for end-to-end VPNs: Generic deployment in an optical testbed environment. *Comm. Mag.*, 44(5):124–132, May 2006.

[53] Reiner Sailer, Arup Acharya, Mandis Beigi, Raymond Jennings, and Dinesh Verma. IPSECvalidate: A tool to validate IPSEC configurations. In *USENIX LISA*, pages 19–24, 2001.

[54] Daniel Senie and Paul Ferguson. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, May 2000.

[55] Gaurav Somani, Manoj Singh Gaur, Dheeraj Sanghi, and Mauro Conti. DDoS attacks in cloud computing: Collateral damage to non-targets. *Computer Networks*, 109:157–171, 2016. Traffic and Performance in the Big Data Era.

[56] Marco Antonio Sotelo Monge, Jorge Maestre Vidal, and Gregorio Martínez Pérez. Detection of economic denial of sustainability (EDoS) threats in self-organizing networks. *Computer Communications*, 145:284–308, 2019.

[57] Lion Steger, Liming Kuang, Johannes Zirngibl, Georg Carle, and Oliver Gasser. Target acquired? evaluating target generation algorithms for IPv6, 2023.

[58] Sudo Null IT News. Creating point-to-multipoint tunnels based on GRE encapsulation in Linux 2.6. Accessed on February 4, 2024 from https://sudonull.com/post/201046-Creating-point-to-multipoint-tunnels-based-on-GRE-encapsulation-in-Linux-26, 2010.

[59] Networking - IP Sysctl Documentation. https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt. Accessed on February 1, 2024.

[60] ZMap IPv6. https://github.com/tumi8/zmap, 2016.

[61] Juha-Matti Tilli and Raimo Kantola. Data plane protocols and fragmentation for 5G. In *2017 IEEE CSCN*, pages 207–213, 2017.

[62] Rene Tio, Suhail Nanji, Ajoy Singh, and Rollins Turner. Layer Two Tunnelling Protocol (L2TP) Over ATM Adaptation Layer 5 (AAL5). RFC 3355, September 2002.

[63] Mathy Vanhoef. Fragment and forge: Breaking Wi-Fi through frame aggregation and fragmentation. In *USENIX Security*, 2021.

[64] Andrew Vladimirov, Konstantin Gavrilenko, and Andrei Mikhailovsky. *Hacking exposed Cisco networks: Cisco security secrets & solutions*. McGraw-Hill Education, 2016.

Appendix

The inwards traffic is the sum of all bytes received by a victim, which includes every even step of the original θ_1, while outwards traffic is the summation of every odd step:

$$\theta_{out} = \sum_{k=0}^{\frac{h}{2}-1} (m - a \cdot (2k+1)) \tag{10}$$

Here, we assume that h, which is the number of tunnelling protocol headers that fit into a single packet, is an even number larger than two. We can work out this formula as follows:

$$\theta_{out} = \frac{h}{4} \cdot (m - a + m - a \cdot (h-1)) = \frac{h \cdot m}{4} \tag{11}$$

The last step is valid because $m = h \cdot a$ where m is the MTU and a denotes the length of the tunnelling protocol header.

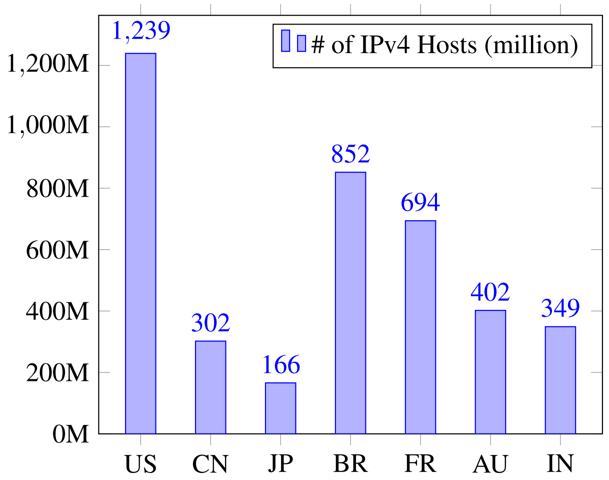


Figure 8: Number of IPv4 Hosts of the most prevelant countries in Figure 4a. The y-axis is the number of IPv4 Hosts. The x-axis is the Country.

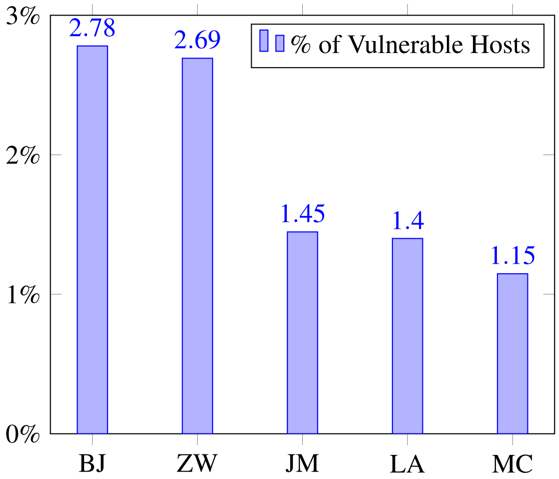
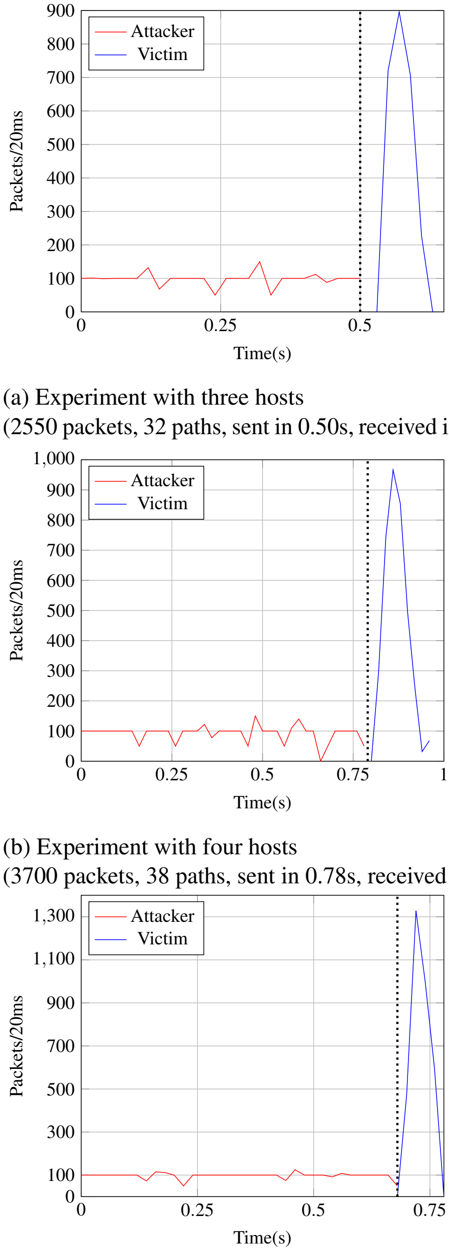


Figure 9: Top 5 most vulnerable Countries based on percentage of hosts that are vulnerable (IPv4). The y-axis represents the percentage of vulnerable hosts over the total number of IPv4 hosts in that country. The x-axis represents the Country.



(a) Experiment with three hosts
(2550 packets, 32 paths, sent in 0.50s, received in 0.078s).

(b) Experiment with four hosts
(3700 packets, 38 paths, sent in 0.78s, received in 0.14s).

(c) Experiment with five hosts
(3400 packets, 33 paths, sent in 0.68s, received in 0.078s).

Figure 10: TuTL attack showing in 20ms buckets the number of packets being sent by the attacker (line on the left of the dotted line) and the number of packets received by the victim (line on the right of the dotted line). Time on the X-axis is synchronised between victim time and attacker time.